ROB316 - PLANIFICATION ET CONTRÔLE

TP n°4 Planication de Trajectoire par l'Algorithme RRT

12 janvier 2020

Gabriel Henrique Riqueti

Victor Kenichi Nascimento Kobayashi

ENSTA IP Paris

Résume

Ce cours présente des méthodes de planification de trajectoire qui ont principalement le but de éviter d'obstacles et chercher un chemin à partir d'un initial jusqu'à le but final. Ensuite, deux méthodes ont été mis en évidence : Rapidly exploring Random Trees (RRT) et Probabilistic Road Maps (PRM). Le premier il s'agit d'une arbre est construit de manière incrémentale à partir d'échantillons tirés au hasard dans l'espace de recherche et est intrinsèquement biaisé pour se développer vers de grandes zones non recherchées du problème. Ils gèrent facilement les problèmes d'obstacles et de contraintes différentielles et ont été largement utilisés dans la planification de mouvements robotiques autonomes.

Finalement, le deuxième, l'idée de base de PRM est de prélever des échantillons aléatoires dans l'espace de configuration du robot, de les tester pour voir s'ils se trouvent dans l'espace libre et d'utiliser un planificateur local pour tenter de connecter ces configurations à d'autres configurations à proximité. Les configurations de départ et d'objectif sont ajoutées et un algorithme de recherche de graphe est appliqué au graphe résultant pour déterminer un chemin entre les configurations de départ et d'objectif.

Question 1

Algorithme	Iterations	Longueur Moyenne	Itération Moyenne	Défaillances
RRT	5000	44.7862	1630.2	0
	10000	41.8748	1786.4	0
	15000	39.0469	1355.4	0
RRT_STAR	5000	31.1524	1887.5	0
	10000	28.1061	1773.9	0
	15000	26.7169	2303.4	0
RRT_STAR_FN	5000	29.3016	725.9	0
	10000	26.2206	1707.8	0
	15000	25.7206	1818.1	0

Tableau 1: Tableau de performance pour les algorithmes RRT

En regardant le Tableau 1 nous vérifions que au fur et mesure que la nombre d'itérations augmente la longueur moyenne diminue vue que avec un nombre plus grand d'itérations, il est possible de généré des chemin plus optimaux. Par contre, le temps de calculs sont plus longues parce que pour faire plus d'itérations il est nécessaire plus de temps pour calculer.

Ensuite, selon les résultats obtenus en faisant les simulations sur Matlab, l'algorithme RRT_STAR*FN présente le meilleur compromis entre le longueur moyenne et le temps de calcul pour les 3 différents nombres d'itérations essayés.

Question 2

Nous constatons qu'à partir de la simulation avec les paramètres par défaut du fichier `FNSimple2D`, le robot ne peut pas atteindre le point finale. L'algorithme essaie plusieurs fois, mais aucune fois le but est atteint et à la fin le rapport statistique affiche un chiffre géant pour la longueur moyenne.

Ensuite, en modifiant le fichier `FNSimple2D_Obst`nous vérifion qu'il est possible d'atteindre le point final. Le Tableau 2 montre le rapport comparatif entre les deux fichiers exécutés pour les mêmes conditions de trajet, point de départ et point final.

Fichier	Iterations	Longueur Moyenne	Itération Moyenne	Défaillances
FNSimple2D	5000	1000000	0	10
FNSimple2D_Obst	5000	111.5078	4014	7

Tableau 2

Pour le premier fichier exécuté (FNSimple2D), le Tableau 2 montre un grand chiffre pour la longueur, mais nous ne pouvons pas faire confiance à ce chiffre puisque le robot ne peut pas atteindre le point final avec ce fichier. En plus, en regardant la dernière colonne il montre que le numéro de défaillances est 10, c'est-à-dire, qu'il a échoué á tous les essaies.

Finalement, en faisant de modifications sur le fichier `FNSimple2D_Obst` nous voyons qu'il est possible de faire que le robot atteint le point final, malgré le numéro de défaillances qui ont eu lieu au cours du processus.

Nous avons mis un oeuvre une algorithme qui est devenu possible que le point final soit atteint par le robot. Néanmoins, il est claire qu'il ne s'agit pas du chemin le plus optimisé.